Workshop

Fakta

Kurs:	Systemutvecklare C/C++ Extended 2024
Klass:	SUVx24
Teknikområde:	Programspråket C, in- och utmatning, funktioner

Learning Target

Programmering i C och C++

Grundläggande begrepp inom programmering.

Skapa källkod med C syntax. Tema funktioner.

Grundläggande in- och utmatning

In- och utmatning av resultat från ett program med hjälp av standardfunktioner i C.

Användning av utvecklingsmiljöer

Använda *Visual Studio Code* (förkortat *VSC*) för att skapa källkod, kompilera och exekvera program skapat med programspråket C.

Grundläggande felsökning

Grundläggande tekniker för felsökning och problemlösning i C

Introduktion

När vi ska lagra data om saker vi vill bearbeta blir det ibland sammansatt information vi ska ta hänsyn till. Hur lagrar vi t ex personuppgifter som namn, personnummer, telefon och mejladress ihop? Svaret på detta kallas **struct** i C. När vi byter till C++ kommer vi i dessa situationer att använda *objekt* för det mesta, men det kan vi inte hantera i C, utan får klara oss med **struct**.

När vi skapar kod är det ganska vanligt att det smyger sig in fel medan vi kodar. Fel i kod kallas en **bug** och vissa fel är väldigt uppenbara (läs ***syntaxfel*** eller ***formella fel***) och fångas upp av kompilatorn, som berättar vad problemet är. Men ibland gör vi tankevurpor och skapar ***logiska fel***. Då kan vi behöva ett verktyg som hjälper oss i vårt detektivarbete att finna felen. Ett sådant verktyg brukar kallas **debugger**, vilket ofta finns för den kompilator du valt. Vi ska titta på vad man kan göra i *VS Code*.

Förberedelse

Skapa en ny projektmapp (katalog) i den projektmapp du använder för kodning, typ:

`C:\Chas\SUVx24\Workshop12\`

Starta VSC på det sätt du tycker är lättast och öppna den nya projektmappen.

Innehåll

Träna på att använda pekare i C

Uppgift 1

Skapa en struct student, tilldela värden och skriv ut innehållet i strukturen.

Att göra:

- Skapa och testkör programmet beskrivet nedan med C-kod i en ny fil kallad **`workshop12-1.c`**

Skapa ett program som

- Deklarerar en **`struct student`** med två variabler:
  - **`roll_no`** som heltal
  - En **`char`** array `name` med plats för **`40`** positioner.
- I huvudprogrammet
  - skapar en student **`s`** med värdet **`123`** för **`roll_no`**
  - och arrayen **`name`** med värdet **`Kim`**.
- Skriv ut respektive värde från structen i två separata **`printf()`**-funktioner.

Uppgift 2

Återanvända strukturen student och ge den ett alias. Använda strukturen på två variabler, tilldela värden och skriva ut från strukturerna.

Att göra:

- Skapa och testkör programmet beskrivet nedan med C-kod i en ny fil kallad **`workshop12-2.c`**

Använd strukturen från **`workshop12-1.c`** och skapa ett program som

- Skapar ett alias på strukturen **`student`** med hjälp av **`typedef`**.
    - Använd structens namn som alias, alltså **`student`**.
- I huvudprogrammet
    - Skapar två student-variabler **`s1`** och **`s2`**.
    - Tar emot uppgifter från användaren genom att fråga efter student 1s **`roll_no`** samt namnet.
    - Tar emot uppgifter från användaren genom att fråga efter student 2s **`roll_no`** samt namnet.
    - Skriver ut på samma sätt som i workshop12-1.c programmet, dvs student 1s roll_no och namn följt av samma sak för student 2.

Uppgift 3

Återanvända strukturen student och skapa en array (lista) av studenter, som tilldelas värden och skrivs ut från listan.

Att göra:

- Skapa och testkör programmet beskrivet nedan med C-kod i en ny fil kallad **`workshop12-3.c`**

Använd strukturen från **`workshop12-2.c`** och skapa ett program som

- I huvudprogrammet
    - Initierar en array med tre studenter:

`roll_no`	`name`
123	Kim
213	Abdul
321	Charles

- Kan genererar en utskrift enligt följande bild:

```
Student 1
Roll number = 123
Name = Kim

Student 2
Roll number = 213
Name = Abdul

Student 3
Roll number = 321
Name = Charles
```

Uppgift 4

Återanvända strukturen student och skapa en godtyckligt stor array (lista) av studenter, som tilldelas värden och skrivs ut från listan.

Att göra:

- Skapa och testkör programmet beskrivet nedan med C-kod i en ny fil kallad **`workshop12-4.c`**

Använd strukturen från **`workshop12-2.c`** och skapa ett program som

- I huvudprogrammet
  - Frågar efter antal studenter att mata in.
  - Skapar en **`struct`**-array med plats för antalet studenter.
  - Låter användaren mata in uppgifterna för en student i taget.
  - Skriver ut innehållet i listan på samma sätt som i **`workshop12-3.c`**.
- Exempel på data att testa med:

`roll_no`	**`name`**
123	Kim
124	Abdul
125	Charles
126	Anya
127	Jenny

Exempel på en programkörning med tre studenter:

```
Enter number of student(s): 3

Enter details for 3 students.

Give roll number and name for student #1
123
Kim

Give roll number and name for student #2
124
Abdul

Give roll number and name for student #3
125
Charles

Student 1
Roll number = 123
Name = Kim

Student 2
Roll number = 124
Name = Abdul

Student 3
Roll number = 125
Name = Charles
```

Uppgift 5

Skapa en struct student, tilldela värden och skriv ut innehållet i strukturen.

Att göra:

- Skapa och testkör programmet beskrivet nedan med C-kod i en ny fil kallad **`workshop12-1.c`**

Skapa ett program som

- Deklarerar en **`struct date`** med tre heltalsvariabler: dd, mm, yyyy.
- Deklarerar en **`struct emp`** med tre variabler:
  - **`no`** som heltal
  - **`name`** med plats för 30 tecken.
  - **`join_date`** med strukturen **`date`**.
- I huvudprogrammet
  - Frågar efter uppgifterna för en anställd:

    `number`, **`name`**, samt **`join_date`** i formatet (**`DD MM YYYY`**)
  - Skriver ut de inmatade värdena från strukturen.
- Exempel på en programkörning:

```
Enter employee number, name, and date of joining (DD MM YYYY):
127 Jenny 27 09 2024

Employee number = 127
Name = Jenny
Date of joining = 27-9-2024
```

Extra: Flytta in definitionen av strukturen **`date`** inuti strukturen **`emp`** som ett aggregat.

- Fungerar det på samma sätt som tidigare?
- Finns det något av sätten att föredra: med eller utan aggregat?